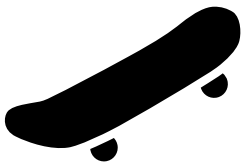

# SKATEBOARD WORKSHOP

Abertay University

**Project Name:** Pong!

**Date:** 19/05/2025

## Welcome & Introduction

Hi there, and welcome to this small workshop we've put together. This workshop is completely asynchronous, allowing you to complete it at your own pace. It should take approximately 10–15 minutes, but feel free to explore more of the codebase if it sparks your interest! If you have any questions at any point, please find a member of the staff team who will be happy to guide you.

You can find a link to the starter code here:
https://github.com/Abertay-University-SDI/skateboard_app_pong_windows.git

## Skateboard & ECS Basics

At Abertay, we've created a small game engine framework called Skateboard, which utilizes an industry-standard pattern known as Entity Component Systems (ECS). While slightly different from Entity Systems in Unity, ECS offers its own method of optimizing the game using a data-oriented approach. In this workshop, we've created a very simple but classic game: Pong! This workshop isn't a tutorial on how to make a game; rather, it's an example of how different layers of a game can enhance various aspects. We'll be adding some polishing elements by incorporating systems into an existing Pong game, improving the player feel and impact.

## Getting Started with Pong!

Let's begin by launching the application. Once you have cloned/downloaded the application from the GitHub link above, open up **Skateboard.sln** in Visual Studio 2022.Launch the application as is. This can be done simply by pressing the Green Button in the Visual Studio window.


When the game launches, you'll see two stationary paddles and a ball in the center that will only move when you press the *Space Bar*.

## Exploring **TutorialScene.cpp**

First, take a look at the file **"TutorialScene.cpp"**. This is where all the magic happens. We build our entities (which are essentially game objects like the paddles, ball, and walls) and then our systems (which contain all the logic for how these entities interact). Entities contain Components that hold data but no logic. This data is accessed by systems, making the code very flexible and modular. We can add and remove some of these systems to introduce additional features to our game, which we'll start doing now.

## Scene Initialization & Entities

In TutorialScene.cpp, the function TutorialScene is the first thing initialized for this scene. We begin by initializing the renderer, which is responsible for drawing everything on the screen. The registry is key here—it stores all our entities and components in a very efficient way. Following this, we create and initialize our entities. The first entities we create are the physics world and camera entities. These are initialized first because we need them to create physics bodies for the paddles and the ball. Next, we create the Right Paddle first, followed by the Left Paddle. We differentiate between the two using a simple Boolean parameter (true/false value) to define which side we are creating the paddle for. Feel free to examine the code of each entity builder function by holding the control key and clicking on the function name. You can always use the back button in the top left to return to the tutorial scene. Eventually, we create the two walls (top and bottom), again differentiated by a Boolean value indicating which side is being initialized. Feel free to go into the function and change the colors of the walls, ball, or paddles. (You can do Task one now if you like.) Now, look at the tutorial scene again—once we've finished creating all the entities in the scene, we then proceed to create a number of systems to handle the logic. Note that the function TutorialScene::TutorialScene() is called only once at the very beginning to initialize our entities, components, and systems.

## Implementing Basic Movement

The *OnUpdate* function is called every frame, which is approximately 60 times per second while the game is running. We start by initializing the physics system, which will be needed by all other systems. Now, let's get to the fun part! Let's uncomment some code—remove the "//" from the front of PaddleSystem. "//" signifies a comment, meaning the code after it will not be executed. Removing these two slashes will now enable the code that was previously greened out. Now, if you run the code again (press F5), you'll be able to move the paddles with the 'W' and 'S' Keys Hooray! We now have a Simple, but fully functional Pong game!

# Adding Enhancements

Now, let’s see how we can take this simple game and improve it!

1. Uncomment the Recharge system and run the application This time, press the ‘D’ or ‘Left Arrow’ to recharge that paddle. Once fully charged, the paddle will multiply the speed of the ball!
2. Once you've tried this, let's enhance the user's visual feedback further by adding some effects. Uncomment the EffectsSystem on the next line and play the game using F5. Notice the changes made. Do you think this improved the look and feel of the game? We generally call these features polish, and they come after the core mechanics have been implemented.
3. Let’s make the ball a bit less static—uncomment the BallEffects and see how that affects your game.
4. That’s all great, but now let’s add some audio to make the game feel alive! Uncomment the audio system and see how the overall feel of the game is enhanced.

## Your Tasks

While I have your interest with this progressive implementation of Pong, let's add some more features and complete some tasks below. If you get stuck anywhere, feel free to reach out to a staff member around! Some of these tasks can be time-consuming, so don't worry at all if you're not able to complete them. There's plenty of scope to learn about this in the years to come.

1. Make the left paddle Green and the Right paddle Blue, reflecting their colors as of the effects.
2. On button press, create an additional ball! Use the 'B' key instead of the Space Bar for this.
3. Play audio "WallHit" when the ball hits a wall. You can do this in the Effects System or create your own WallSystem, similar to the PaddleSystem.